IN THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Previously Presented) A method for handling an interrupt, the method comprising:

receiving an interrupt request corresponding to a particular interrupt;

upon receiving the interrupt request, substituting a vector corresponding to a group of interrupts with a vector corresponding to the particular interrupt, wherein said substituting does not require performing a polling operation; and

jumping to a service routine corresponding to the particular interrupt responsive to said substituting.

2. (Original) The method as recited in claim 1 further comprising obtaining the vector corresponding to the particular interrupt from a lookup table.

3. (Original) The method as recited in claim 2 wherein the vectors corresponding to a group of interrupts and the particular interrupt are addresses.

4. (Original) The method as recited in claim 3, wherein the lookup table includes an address corresponding to the group of interrupts and a plurality of addresses each corresponding to one of a plurality of particular interrupts, and wherein each of the plurality of particular interrupts has a corresponding service routine.

5. (Original) The method as recited in claim 4, wherein the address of the particular interrupt is the address of the group of interrupts combined with an offset address.

6. (Original) The method as recited in claim 5, further comprising

trapping an interrupt vector address responsive to said receiving the interrupt request and substituting the interrupt vector address with a jump table address;

executing a first jump instruction, wherein execution of the first jump instruction causes a jump into a jump table at the jump table address, wherein an entry associated with the jump table address includes a second jump instruction; and

executing the second jump instruction, wherein said executing the second jump instruction causes a jump to the service routine corresponding to the particular interrupt.

7. (Previously Presented) The method as recited in claim 5, further comprising:

executing a first jump instruction, wherein said executing the first jump instruction causes a jump into an entry in a code table indicated by a code table address, wherein the entry indicated by the code table address includes a second jump instruction and an associated jump table address;

trapping an interrupt vector address responsive to said receiving the interrupt request and substituting the interrupt vector address with the code table address;

executing the second jump instruction, wherein said executing the second jump instruction causes a jump into an entry in a jump table indicated by the jump table address, wherein the entry associated in the jump table address includes a third jump instruction; and

executing the third jump instruction, wherein executing the third jump instruction causes a jump to the service routine corresponding to the particular interrupt.

8. (Original) The method as recited in claim 1 further comprising:

setting a first bit in a first register responsive to receiving the interrupt request;

ANDing the first bit with a second bit, the second bit stored in a second register, wherein the second bit corresponds to the first bit, wherein said ANDing

produces a third bit, wherein setting of the third bit is indicative that the interrupt request corresponding to the particular interrupt was received and that the particular interrupt is enabled.

9. (Original) The method as recited in claim 8 further comprising resolving a priority of the particular interrupt within a plurality of interrupts.

10. (Original) The method as recited in claim 1 further comprising executing the service routine responsive to said jumping.

11. (Original) A processor comprising:

an interrupt source register, wherein the interrupt source register configured to store a plurality of bits including a first bit corresponding to a particular interrupt, wherein the first bit is set responsive to receiving an interrupt request; and

a read-only memory (ROM), wherein the ROM is configured to store a vector corresponding to a group of interrupts and a vector corresponding to the particular interrupt;

wherein the processor is configured to substitute the vector corresponding to the group of interrupts with the vector corresponding to the particular interrupt and further configured to jump to a service routine corresponding to the particular interrupt.

12. (Original) The processor as recited in claim 11, wherein the processor includes an enable register configured to store a second bit, the second bit corresponding to the particular interrupt, wherein the second bit, when set, is indicative of the interrupt being enabled.

13. (Original) The processor as recited in claim 12, wherein the processor implements an AND function configured to produce a third bit based on the first bit and the second bit, wherein the third bit, when set, is indicative that an interrupt request corresponding to the

particular interrupt has been received by the processor and that the particular interrupt is enabled.

14. (Original) The processor as recited in claim 13, wherein the processor includes a priority encoder coupled to receive the third bit, wherein the priority encoder is configured to resolve a priority of the particular interrupt within a plurality of interrupts.

15. (Original) The processor as recited as recited in claim 14, wherein the processor includes a lookup table coupled to the priority encoder, wherein the lookup table includes an address corresponding to the group of interrupts and a plurality of addresses each corresponding to one of a plurality of particular interrupts, and wherein each of the plurality of particular interrupts has a corresponding service routine.

16. (Original) The processor as recited in claim 11, wherein the vector corresponding to the group of interrupts and the vector corresponding to the particular interrupt are addresses.

17. (Original) The processor as recited in claim 16, wherein the vector corresponding to the particular interrupt includes an interrupt group address and an offset address.

18. (Original) The processor as recited in claim 17, wherein the processor is further configured to:

trap an interrupt vector address responsive to receiving the interrupt request, and substitute the interrupt vector address with a jump table address;

execute a first jump instruction, wherein execution of the first jump instruction causes a jump into a jump table at the jump table address, wherein an entry associated with the jump table address includes a second jump instruction; and

execute the second jump instruction, wherein said executing the second jump instruction causes a jump to the service routine corresponding to the particular interrupt.

19. (Previously Presented) The processor as recited in claim 17, wherein the processor is further configured to:

execute a first jump instruction, wherein executing the first jump instruction causes a jump into an entry in a code table indicated by a code table address, wherein the entry indicated by the code table address includes a second jump instruction and an associated jump table address;

trap an interrupt vector address responsive to receiving the interrupt request and substitute the interrupt vector address with the code table address;

execute the second jump instruction, wherein executing the second jump instruction causes a jump into an entry in a jump table indicated by the jump table address, wherein the entry associated in the jump table address includes a third jump instruction; and

execute the third jump instruction, wherein executing the third jump instruction causes a jump to the service routine corresponding to the particular interrupt.

20. (Original) The processor as recited in claim 11, wherein the processor is configured to execute the service routine in order to resolve the particular interrupt.